International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 3 Issue 10, October-2014, pp: (80-90), Impact Factor: 1.252, Available online at: www.erpublications.com

An insight on the advancements in Object Oriented Programming-Teaching Methodolgies, Technology Inclusions and Applications

Anuj Rastogi[1], Priyam Saxena[2]
[1,2]Systems Engineer, Infosys India Limited, Hyderabad, India

Keywords: Ada, Design, Modules, Object Oriented Programming, Tools, Teaching Methodology.

INTRODUCTION

When the software industry was still in its infancy then the major attention of the newly evolved software was to deal with the then-prevailing problems of humans, as over work etc. But, with the growing age and thus, widening ratiocination of the humans towards the domain of software, it was realized that in the quest of attaining the goals the entities for whom the software was developed have lost their identity, in the course. Also, if the ulterior goal of software was to extend the benefit to the end consumers then, their inclusion as an integral part into programming was necessary. With this, in the dusk of the century, Object oriented programming (OOP) concept was proposed whose cynosure was less likely the goal but was the entity. With the belief that the information about the entity can equip a programmer to perform indefinite operations on it, has emanated and bolstered the pervasive use of OOP. Thus, through the passage of time OOP has found its way to become one of the most acceptable programming and design techniques. This paper is, thus, an attempt to project the power of OOP through various technological innovations incorporated into it and through the benchmark applications. The paper starts with the ideas prevailing to further expand this pragmatic programming technique to the students and its aspirants in the most novel ways.

TEACHING METHODOLOGIES

Some novel teaching methodologies have been developed and have evolved through experience, for effectively teaching the concepts of OOP. These are discussed with an overview in this section. JAVA, Object Oriented Programming (OOP) is an intricate syllabus to teach. There needs to be a generic core structure to solve the imbroglio of what to explain first and where to begin with while teaching green horns. In the [1]-[3] a generic course structure was presented suggesting a curriculum. Firstly, the basic of OOP needs to be presented. Since majority students coming to JAVA have a prior C experience, hence, the language technicalities need not to be focused upon. The main objective should be to exhume the student’s mindset from procedural programming and inhume it into Object Oriented thinking. In this phase a primitive knowledge of data types should be presented. Also, a knowledge of predefined classes such as String and Math etc. providing forward the concepts of class and object should be explained with the difference between a class in JAVA and modules in C. classes in JAVA are real world objects.

Hence, the differences should be told. Moreover, every time a class is used, the default value of variables become zero, which is not likely the case with C. Further, the extensive use of properties inside the class should be explained e.g. constructor inheritance and polymorphism. In [1]-[3], this study was divided into 2 modules. One module defining class, which included explanation of JAVA classes, definitions of constructors and also access controls and encapsulations was propounded. Second module collated concepts of inheritance constituting: (1) calling a constructor of super class (2) overriding and hiding methods (3) overloading methods. Hence, [1] acted as a generic curriculum, winnowing the doubts of teaching JAVA, OOP.

Currently, the focus of teaching industry is to automate the teaching process and further develop artificial intelligence in these systems. This is to help and provide verdant students personal and intelligent attention through such systems. It is only through automated modules that students can get individual attention. In [4]-[6], one such tutoring application was discussed having expert system. In this application code was developed using a generic architecture and expert modules purpose was to debug and assist the novice programmers to learn the concepts of OOP. For this, the expert systems module was provided with artificial intelligence. Based on this and keeping in view that the present application modules can be incorporated in more sophisticated developments, the generic architecture has 10 modules.

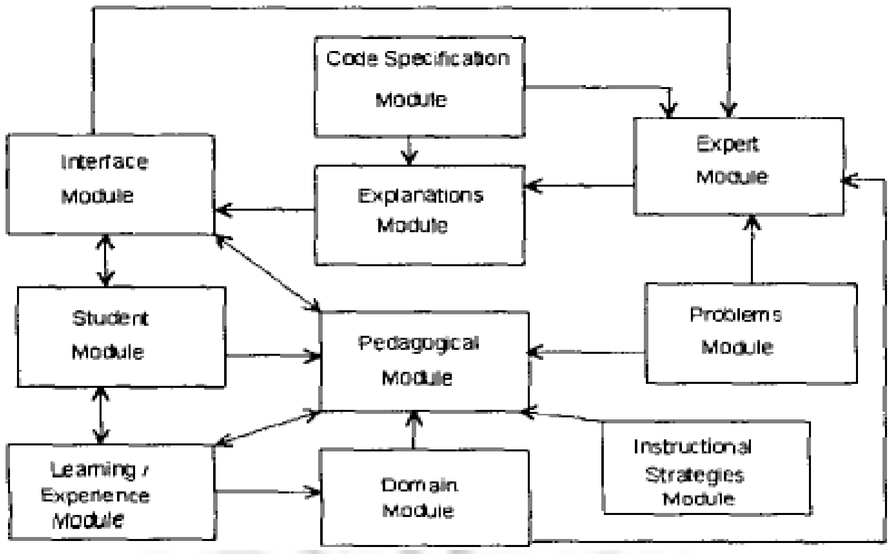


Figure 1: Generic structure proposed

Of all modules presented, expert module was the most important module. With following expected functionalities, it should have artificial intelligence to tackle the doubts of neophytes and also, debug the codes for OOP provided to it. Further, it should provide a necessary solution to the verdant programmers. Also, it should be able to comprehend the problem and hence, extract to the maximum the solution by reusing the existing classes. For sensing such functionalities and promoting OOP understanding, and thus promoting artificial intelligence, it should have knowledge about (1) the entities in OOP and objects as well as relations among them (2) methods applied to entities (3) inputs to the problems and hence, the necessary output demanded. Based on these design inclusions as soon as it receives a problem statement, by alienating the input from output, it constructs OOP algorithm and also helps to debug the code developed. Also, a neural network has been incorporated into system for future enhancements. For designing the algorithm and deciding the methods that need to be incorporated into a class it uses a generic programming. Moreover, different mechanisms have been introduced into the expert modules to deal with deterrence programming. In future it is looked for, that this software can also produce entire programs, converting algorithm's functionalities. A separate component will be required for this purpose.

Today, majority of the students are involved in playing games. A survey shows that the students now-a-days can play up to 10,000 hours of games a year. Hence in [7]-[8], learning OOP (Object Oriented Test) through games was proposed as a methodology. This was promoted with the thought, that, let students learn from the thing they enjoy. Also, since the traditional teaching methodologies are one-way and currently, students are involved in individual studies. Hence, games can be effectively used as they promote two way interaction, making learning active. Also, with the advancements in game technology, it is appreciable to use them in advanced learning. A number of games already exists for educational purposes. In such games, the functions of the games are inspired by the learning and exploring them indirectly means driveling into the subject. In OOP learning through gaming and promote progressive learning a story telling strategy was followed. The overall game was a mini puzzle, based on the learning from the story-telling and to attain next stage the present needs to be crossed. From research it was found, that, for successful teaching through games it must have an interactive interface and an inclination towards learning, with a zing of entertainment. The research approach followed was the action research approach, for it concentrates the efforts on the subject with a hope of flexibility in the result in future.

Hence, it contributes both to the people's demand and goals of social science. In constructing the design of the game, first, the requirements of the user were identified and then based on them objects and attributes were determined. After this, the interaction between these objects in the game was analyzed through UML (Unified Modelling Language). The entire flowchart on how the design was proceeded with, is as in figure (2) (Left).The purpose of Game Object Mode (GOM) is to help the designers to understand the learning objectives that need to be a part of the game. Persona Outlining Mode (POM) was to identify the actors and objects that will be involved in story-telling and Game Achievement Model (GAM) helps to achieve the design interface. Because, the game design was such that, through it knowledge-transfer and hint-availability both were bolstered hence, it allowed the children to mull over things. The final game mode has five levels. They had deterrence in the form of terrorists. By crossing those deterrence only, a person can proceed. In doing all this, indirectly OOP concepts were also taught. Hence, teaching in five levels was promoted. First level was object and class test, second level was if-else conditional statement test, third level was an array test, fourth level was OOP principle test having principles of inheritance, encapsulation etc. and fifth level was a test of polymorphism. Moreover, while designing since both education and entertainment were to be introduced into the game, hence, the best design approach selected was Amory and Seagram Framework.

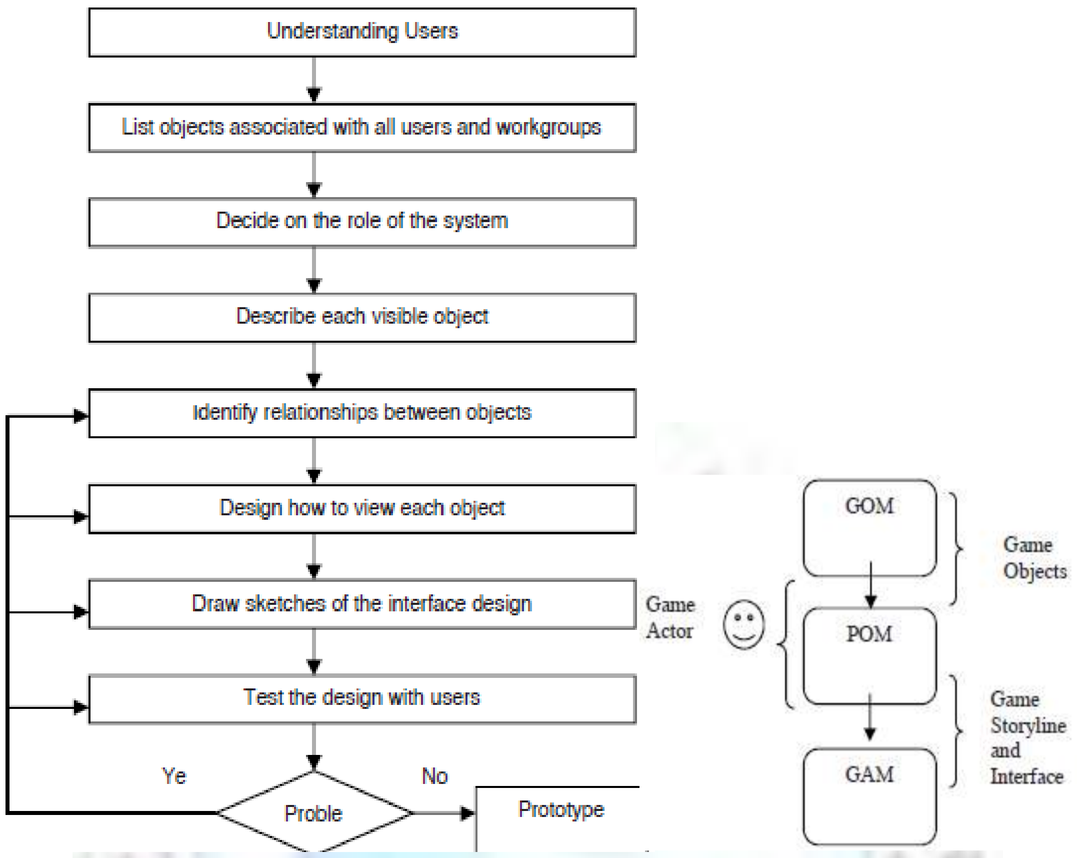
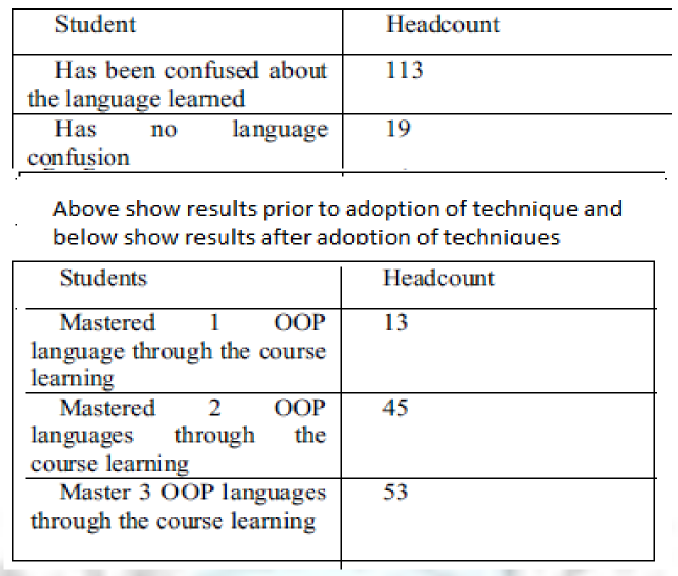

Understanding Users
List objects associated with all users and workgroups
Decide on the role of the system
Describe each visible object
Identify relationships between objects
Design how to view each object
Draw sketches of the interface design
Test the design with users
Ye
No
Proble
Prototype
GOM
POM
GAM
Game Objects
Game Actor
Game Storyline and Interface

Figure 2: Flowchart of Object Oriented Software Engineering Methodology (Left) and Game Construction Framework (Right)

In [9]-[10], Smalltalk was defended to be one of the best language for learning (Object Oriented Programming) OOP and acquiring Object Oriented mentality. Since, Smalltalk is a language to develop graphics so during studies student got a lot of exposure to project their creativities. In the course of learning OOP, JAVA and C# are considered to be the best and most profound language. But due to their huge library the OOP training in these force the students to pay more attention to technicalities than developing an object oriented orientation. The following things, thus, in Smalltalk made it adorable and ideal for developing OOP mindset among students.

First was Object and Messages in Smalltalk. In Smalltalk because of its confined operational area, there are only objects and messages. All things are to be developed with this support only for they are the only technical present. Hence, students are forced to think in Object Oriented way and this develops an attitude. Secondly, because of generalities in Smalltalk and no data-types a lot of complexity is reduced allowing students to focus on the concepts rather than language niceties. Lastly, in Graphical User Interface (GUI) design the developer has no other option than to define different classes to support various screen objects. This obtrusion dealing with objects develop an object aptitude. Thus, Smalltalk provides a way to learn OOP fundamentally. A similarities in class libraries with C# and JAVA make students transcend limits and understand these purely OOP concepts.

In the normal teaching curriculum, while OOP (Object Oriented Programming) is taught the major focus of teaching is to make the students learn the language, without delving into the basic concept of OOP. In majority cases, as the students proceed to learn other OOP language then they tend to get confused with the syntax and they start developing a propensity towards the earlier learnt language. This confusion, among learning different language, is a projection of a lack of common concept of Object Orientation, which is at the basic level of each language. In independent learning of languages, students tend more to get involved in syntax and nuances of the language than trying to relate OOP concept with language. For the same in [11]-[12], a new approach of teaching OOP was propounded where in it was told that the teaching focus should be more towards Object Oriented mindset and not towards understanding and delving in the nuances of the language. In this OOP thinking all three languages C#, C++ and JAVA should be taught together related to each other and on a common platform of object orientation. This organizes the teaching and moreover removes confusion as the focus of learning digress from language specific to concept specific. Also, in [11], coding standard based learning was promoted, so that students at a very early age learn to develop maintainable programming practices. Results prior to adopting the methodology and the difference after adopting it are shown in the figure (3).



Student	Headcount
Has been confused about the language learned	113
Has no language confusion	19

Above show results prior to adoption of technique and below show results after adoption of techniques

Students	Headcount
Mastered 1 OOP language through the course learning	13
Mastered 2 OOP languages through the course learning	45
Master 3 OOP languages through the course learning	53

Figure 3: Results after adoption of technique

TECHNOLOGY INCLUSIONS

Since, the evolution of OOP multiple new techniques have been imbued or obtruded into it for the sake of improving the utility or either for further providing privilege to the people. A few of those techniques that have left a lasting impact on the OOP, over the last decade, are discussed. OOP is a new programming technique when compared to conventional programming. For OOP, we need to determine the important elements. In [13], one prototype OOP tool was built based on heuristic rule for accurate determinations of entities like class, variables, function/methods etc. This tool proved excessively useful for the college students, as without determining the entities one cannot go further with OOP. Also, this gave them an idea about how to determine an object. In OOP's each class corresponds to objects and due to inheritance an object can represent multiple classes. Thus, each class representing an object is a real world representation of an object in the core of programming. Hence, real world objects can be represented by classes in programming. In OOP (Object Oriented Programming) an object can represent multiple classes such as a super class, multiple super class. Hence, there is a multiple participation of an object with classes.

In the real world scenario it is hard to comprehend which specific class does the object belongs to. If this problem of uncertainty is dealt then multiple inheritance is also supported by partial membership of an object. At times it is a real world requirement that an object can acquire attributes of a different object. Then there is a dire need of partial membership, for with this an object can acquire, with rationale, the properties every class. Since, OOP does not support uncertainty and strong declarative semantics which is the basic for logic programming, hence, it cannot inferentially make out what properties of which class need to be attained in partial membership. Alone, it cannot support multiple inheritance. Also, when object has inherited classes then also there is uncertainty, since, an object belongs to all classes. For such ambiguous situations in [14]-[15], an interconnection of OOP with FRIL, strong logic language with supporting Fuzzy Objects, was proposed. Pure logic programming can be used to manage program, properties and behavior. In OOP, an object has its own data values for each field. In logic programming instead the methods define rules and fields are similar to facts. Thus, with the inclusion of strong logic with OOP uncertain situations can be dealt with. FRIL is a language with strong capabilities to handle uncertainty. Also, Fuzzy Browser has been developed using it. Also, FRIL can handle uncertainty between relationships of data items. A fact represents relationships between two or more objects. For this, FRIL allows conditionally true statements. FRIL resolves uncertainty with the help of 3 rules: (1) basic rule (2) extended rule (3) evidential rule. In evidential rule even if certain traits of an object are pen chanted towards properties of some class, then, it resolves the uncertainty as to which class the object belongs to, by inferential deduction. This extended rule allows a conclusion to be drawn even if all conditions are not met. For managing intuitive interface FRIL uses dialog compilers.

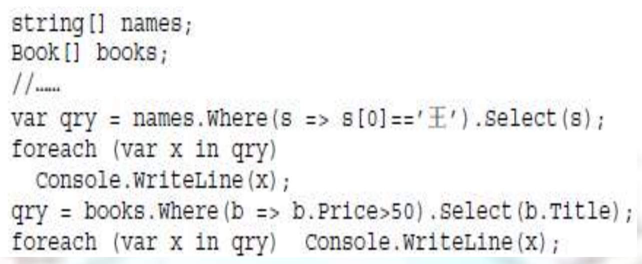
Thus, with the inclusion of FRIL in OOP, which particular class properties the object should attain, while belonging to different classes, can be deduced. With the resolution of this problem a concept of FUZZY object comes into picture which competes with the challenge of uncertainty of object-class participation and multiple inheritance. A fuzzy object is a member of more than one class, having a relationship in each class. This is unconventional to OOP where in each set of Object correspond to one class, having properties of only that class. FRIL helps to determine at runtime that which class Object should map to. For doing this a class membership method in class is placed and this dynamically decides, based on properties object wants to acquire, as to which class it should retain the membership with. Based on the above proposal a rule was drawn in [14], where in if Xs and Xc are class membership functions of super-class and class S and C respectively then, with X elements Xc ≤ Xs. FRIL incorporates FUZZY sets as fundamental data types and with its inclusion in OOP this confusion of object to class mapping and versatile nature of object was increased.

In the languages such as C or C++ unlike Ada or other languages, there are no additional features that can ensure time security or error checking. Also. There is no such feature that supports concurrency. This concurrency issue is left in both C and C++ on the mercy of Operating System. But in case of embedded systems, such issues need to be handled in by the language. Hence, the use of OOP in embedded system need the intrusion of concurrency properties in the language. For this in [16] – [17], an extension of C++ called ClassiC was proposed. The ClassiC is a language with minor extension in C++ to support process concurrency. The emergent specifications for process concurrency were revealed when the new version of Ada came up with the requirement of integrating OOP with concurrency. Ada 9X supports OOP in the form of tagged the records. Although, different from "task" construct, this providing concurrent processes is the style of Ada. Thus the result of integrating concurrency with C++ has been a language after which while designing the system also, the concurrency of objects need not to be mulled about. Since, the language at every point supports it. In classic the process is also understood as an object, having strands of processing associated with it, hence called active object. Since the process is also an object, so all principles of OOP e.g. abstraction, encapsulation etc. apply. The process is none but the extension of a class construct. In addition to normal access specifiers, it has one more by the name "entry" which promises mutual exclusion. In ClassiC process, there are multiple constructors. After instantiation, by process object, constructor returns control to the object because of "coreturn" statement and at the same time the child process keys on running because of constructor. The unusual execution is to provide active class inheritance.

Also, the multiple constructors allow, many process functions to be defined in one class itself. For any process inter-process communication is highly important. An object is something on which operations are performed. Hence, in classic process, since process are treated as objects, are something which should support operations, inter-process communications etc. For providing inter-process communications some measures in the class are added with function which can be called by other objects ensuring mutual exclusions. In ClassiC "accept" statement implies a process is blocked for communication with others. Because of the blocking mode there should also be a statement that ensures when the process is ready. For this select statement based on Hoare's, select instruction in CSP. There can be multiple process instances, which can also communicate using these member methods or functions, having functionality which allows the two process objects to exchange information. The process inheritance is similar to OOP. The conductor where an active class is derived from inactive class is also described. In such a case the priority needs to be declared. If such has not been declared, a possibility of concurrent access to the member of the base class exists, without mutual exclusion. Also since every languages now a days support interface events, the same is also the case with ClassiC which supports event handling, with the help of interrupt class. The vector number as parameter is fed to its constructor and has a virtual int operator ++ ( ), (entry). This method is defined as virtual so that it can be seen promoting inheritance and thus expressing the overridden.

For the scope of interrupts, this is also provided with default construct, to permit derivation of multiple interrupts without interrupt handling. In ClassiC there is no explicit priority handling in a particular process. Every process in child class has the priority of parent which changes by explicit system calls. Since classic is only a basic extension of C++, hence, the same compiler can be used. The additional keywords coreturn, sect, entry, accept, when and associated grammar are incorporated into parser. The classic provides facilities of OOP with concurrency.OOP (Object Oriented Programming) is the soul of all major applications now a days. It focusses on the object rather than the goal. The very basic OOP, thus, lack somewhere a skill of problem solving. For the same, object oriented programming is integrated with constraint logic programming to give birth to OOCP (Object Oriented Constraint Programming). In the OOCP, determining the methodology for designing the constraint part is a tedious job. Hence, no such standard methodology exist as to design the OOCP with the help of guidelines. In [18]-[19] for the first time a methodology for designing the OOCP was proposed. It was told that in OOCP the major design part still remains about OOP designing and analysis and only a small augmentation in its design can yield the design methodology of OOP. Thus, in [18] the OOCP methodology was proposed in there it was shown that only an additional 2-steps were required to make OOP system constraint oriented. These were: (1) automating design with constraints and variables (2) define goals to achieve. In the first step, once the Object Oriented design has been done in each case the unknown variables and constraints are selected and they are assigned looking into the bondages that no law violates. These constraints variable are one which are necessary for further action of object or which either determines the work an object has to undergo or its function. If in case they cannot be determined, then the entire class should be replaced with the one where they can be determined, like in case of hospital where nurse are assigned shifts. Now, which shift the nurse has to be assigned is a constraint variable, depending upon a number of conditions. Once these constraints variable are determined, it is clear as to what goal a particular object has to achieve. For this, in modelling, a state diagram is borrowed to represent OOCP goals. For the action of this variable, we instantiate the variable in the class demanding for an action.

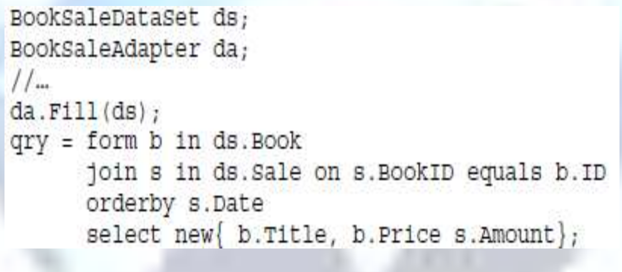
Hence, the designing technique enhances the OOCP uses which are directed both towards object and goal. The main aim of the constraint based variable is to identify the relationships between the constrained variable and domain classes within the model.In normal curriculum OOP (Object Oriented Programming) and Data Base Management System, or SQL (Sequential Query language), are taught separately. Thus, it is hard to relate and understand the integration of two different programming systems. Although, C# or ADO.net extend the capabilities to access the database by integrating the SQL capabilities but still, it is hard to understand them in an integrated way. For the same, in [20]-[22] a technique of first involving LINQ with OOP to understand the relation between OOP and how to fetch data was proposed. With the knowledge of LINQ and small transition in coding style, Relational Data Base Management System (RDBMS) becomes easier to understand and use with OOP. For the same in [20], a curriculum of how to teach LINQ with OOP was proposed. For this LINQ curriculum was divided into three groups. When LINQ is used with objects, here the basic purpose of the query is to fetch the objects of the same type and group them as e.g. in the figure (4).



```
string[] names;
Book[] books;
//……
var qry = names.Where(s => s[0]=='王').Select(s);
foreach (var x in qry)
  Console.WriteLine(x);
qry = books.Where(b => b.Price>50).Select(b.Title);
foreach (var x in qry)  Console.WriteLine(x);
```

Figure 4: Query to fetch objects of same type

With the selection and grouping other LINQ statements and their syntax can be taught such as Selectmany, OrderBy/ Orderby Descending, Group By, Join, GroupJoin etc. When LINQ is used with ADO.net, here LINQ is used to fetch data from heterogeneous sources. It is to be noted that the teaching should be limited to storing the data temporarily in data table or sets. Also, they should be taught as to how to load that data set data. An example of fetching using LINQ is in figure (5).



```
BookSaleDataSet ds;
BookSaleAdapter da;
//…
da.Fill(ds);
qry = form b in ds.Book
      join s in ds.Sale on s.BookID equals b.ID
      orderby s.Date
      select new{ b.Title, b.Price s.Amount};
```

Figure 5: Fetching in dataset using LINQ

Also, in this phase focus should be on defining entity classes and mapping them to corresponding database table, writing LINQ equivalent statements to fetch data from database tables. Although, every class and LINQ queries come under two different programming as query language comes in Data Access Layer (DAL), but an understanding that every entity maps to a row in the database table has to be made. When LINQ is used with XML, here, the LINQ working in XML files need to be understood through a collection of unified APIs. Here the LINQ in the program is first extended to objects program by adding binary files. Then by transforming the binary files XML the entire query in OOP can be transformed to XML and hence can be used by any other language. Because of the resemblance of LINQ with T-SQL, both in syntax and function, it is easy to understand T-SQL. Hence, OOP integrated with LINQ provide better understanding while integrating T-SQL with OOP

APPLICATIONS

The OOP being the language that promotes reusability, user identity and of course ease of use has found its way to the end users. Today, a number of applications have this at their base. Some of the benchmark applications of this technique are discussed as under.

EMC is a state in which it is supposed that, each electronic equipment perform individually without the presence of the magnetic field of any other electronic equipment in the proximity. With the growing electronic traffic it is but a dire need to preserve the independent functioning of instruments. For this, EMC tests are done. But with the growing complexity of systems and a diurnal need of accuracy, automation of such EMC testing needs to be done. For this in [23]-[24], a technique was proposed for automating the EMC test using Object Oriented Programming (OOP) idea in the designing of EMC test software. 5 objects were propounded which corresponded indirectly to five independent stages. They are: (1) test template object (2) instrument objects (3) apparatus object (4) graph object (5) report object. The test template object encapsulates methods which decides the parameters and procedures of the testing of a particular item. The instrument that are needed to be involved in that process are controlled remotely through instrument objects and the result of that is controlled by apparatus object. The results be getted is controlled by graph and report object.

The test template object has three major functionalities. It encapsulates basic test information. The other controls to frequency that needs to be set up for test. This method of selecting frequency band based on test modes is provided by GJB152A. There are 2 test modes namely: receiver and spectrum mode. In receiver mode, we use mid band while in spectrum mode we use resolution band. Further, one method deals with setting up instruments and parameters for test. Also, the instrument object encapsulates methods to remotely give commands to instruments. Whatever instruments are to be used during testing are provoked through this object. It has VISA instructions to do so. Since, any instrument can have multiple VISA instructions so, the VISA instructions are encapsulated based on functions. Thus according to functions instruments can be divided into segments. Moreover, there is an apparatus object which controls those apparatus, effecting the results of test. The difference with instrument object is the fact that no remote control is allowed in this. The apparatus such as antenna, sensors, attenuator etc. are controlled. The main function of apparatus object is interpolation of transmission coefficient. If corresponding to A1 transmission coefficient, frequency f_1 is allowed and with A2, f_2 is allowed. Then, with A and frequency f, where $f_1<f<f_2$ then the expression for A acquires:

$$A = A1 + \frac{\lg f - \lg f_1}{\lg f_2 - \lg f_1} \cdot (A2 - A1)$$

Figure 6: Expression for "A" (Transmission Coefficient)

Finally, graph and report objects encapsulates methods to plot the test results effectively. Thus, with the automation and redressing manual interface precession was achieved.

In fire protection one has to deal with challenging situations. For different sorts of fire situation there are different techniques and every technique needs to be implemented with accuracy. Moreover this installation is also actively involved in design and installation of fire protection system. Thus, an exact record of all these systems need to be maintained. Earlier they were managed manually but it requires a lot of man force and finance. Hence, an approach to make this process economical with both money and time was needed. Since, using component in the protection system has independent attributes and real world things are most effectively presented with the help of Object Oriented Programming (OOP). Thus, the need for automation was accomplished involving the concepts of OOP, within the system.

In [25]-[26], the same was explicated in an elaborated way. The ground shaking reality of the system divulged that although, the systems design, while automation, was developed by object oriented perspective but are non-object oriented language was used to translate design into computer's code. The precision of OOP's helped significantly in alleviating inefficiency and inconsistency. For the same, OOP system developed for automation was called PROTECT (Production, Resources, Order, Enquiry, Technical specifications, Estimation Computer Oriented Tool). The tool was a centralized tool thus, all procedures linked with it were fully automated. Since, it complies with mutual and international standards the system was widely accepted. The capability to reduce time and labor because of automation helped it increase efficiency. Since the system tools were earlier designed with a non-OOP language. Hence, with increasingly complex situations it became hard to maintain. The recourse coerced the people to take system to OOP and thus it was reanalyzed and re-implemented without changing any functionality.In system architecture, PROTECT contains entire information on how to react on various situations. Hence, the input to the system is given through PROTECT and based on situation the flow of stimulus promotes other system to react example- stating whether to open 2 nozzles or 3 nozzles. In the object oriented design of variegated components in 5 organizations, 3 aspects were dealt with as: Object model, dynamic model, functional models. Object model describes the object classes and its relationships. The functional model describes the values and the functions associated. Thus, in the first step of design the various sub-systems were produced with the system and their common aspects were shared. In the second step, implementation of such systems was done. In this, further details in object class were added and the implementation decisions were taken. Each functional model operation is translated into algorithm.

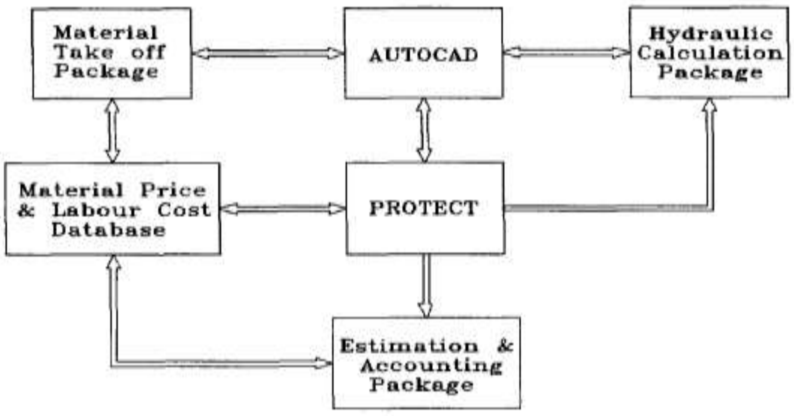
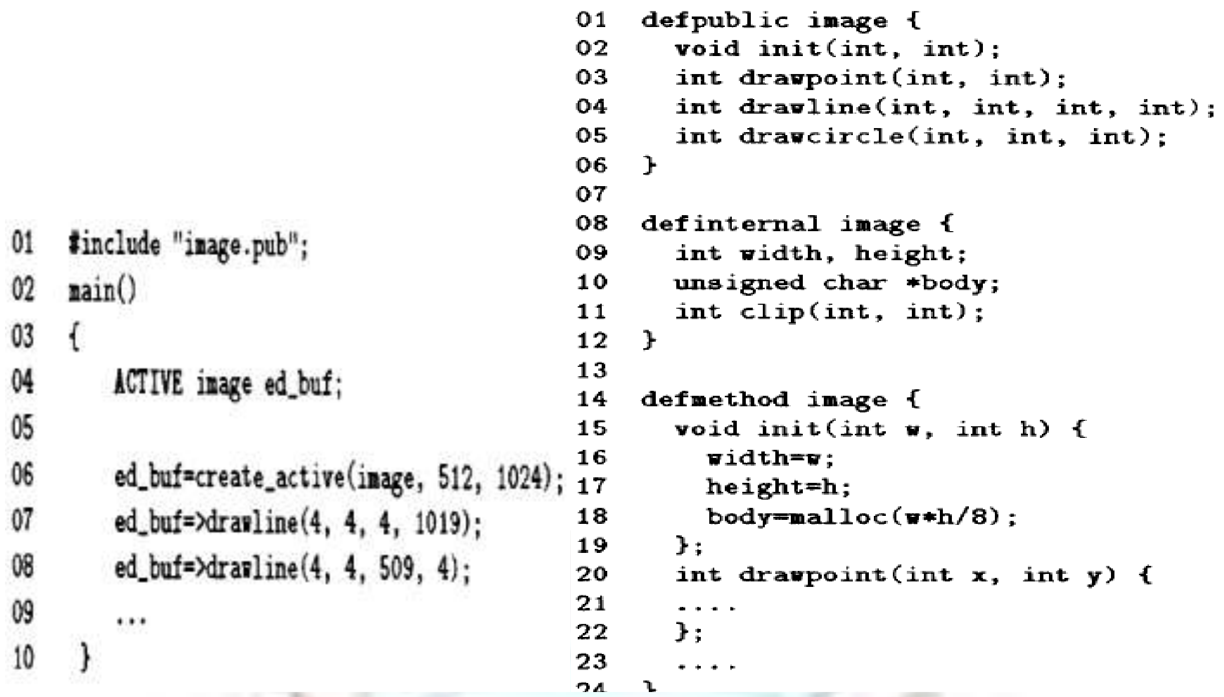

Material Take off Package
AUTOCAD
Hydraulic Calculation Package
Material Price & Labour Cost Database
PROTECT
Estimation & Accounting Package

Figure 7: System architecture after PROTECT installation

In designing, a perfect encapsulation is also ensured. Even though the designing of the system was done in OOP, implementation was done in non OOP. This was to avoid developing the code from scratch and proceed on already existing system. For translation of design to code, FORTRAN was chosen. In the first step, each object with its attributes and functional relationship, as in old system, was mapped to new FORTRAN functionality. Further, new data structures were built and linked to produce PROTECT in OOP.

FORTRAN has several special features, firstly, it supports complex data constructs, helpful for translating classes into data structures. Also, it supports inheritance. Although, FORTRAN does not support encapsulation but this can be rectified using multiple entry points. An OOP makes the program easy, reusable, expandable and efficient.In network data communication, an RPC (Remote Procedure Call) model developed by Birtel and Nelson is used currently. This RPC supports abstraction between client and server and hence, simplifies the server applications. But in intricate situations since, in RPC to utilize the server method a procedure call to the server is made hence, unwantedly the client has access to many methods of server and thus, encapsulation and abstraction is compromised. This also exemplifies the fact that RPC is not built upon a totally Object Oriented Language but the programming style is procedure oriented. To overcome this drawback of abstraction in RPC used for communication, in [27]-[29] an ROI (Remote Object Invocation) technique was proposed. This ROI was first implemented in the MONARC system, an OOP support for UNIX. In ROI an active object approach was utilized. This ROI was totally developed on OOP and hence, complete server and client abstraction was provided. ROI invokes methods through invocation object instead of procedure calls and this assures complete abstraction revealing only method, the one which is called. In ROI model the client invokes an object at the same end. Hence it is required that this object contains server methods to serve the clients. Here comes the concept of active objects. This contains the methods and a set of data to exchange message with the client. The ROI communication is performed in messages in the stream and are statistically defined. In the methods with a return value the values are transferred by call-by-value mechanism. To support ROI, the following extension should be provided by the language of the platform: (1) possibility for creating server class (2) possibility of instantiating that class from client end (3) static typing of communication protocol.

In addition to this, when the Server side compilation is done, class is converted into server process, instantiating which a client can exploit its methods. Also there is a requirement that a communication should be done transparently for which sub-routines are used. The stub includes code for: (1) main loop waiting for instantiation (2) a dispatch, to dispatch a selected method (3) sending back the result.The communication protocol consists of the request method from the client and the reply method, from the server. Every request message has first, a request header, then a message sender arguments. The reply consists of the reply header and the result from selected method. In such a model it is possible that the client at different location can use the services at the server end, as they are separate instances. But the problem arises when two instances from the same client are created. To avoid that in MONARC system concurrency in unique client is not allowed. Only after processing of one request can other request be processed. For this there is a port number in every machine, through which every machine is attached to the server. This is sent with the invocation for distinguishing between two instances from same or different machines. This Object sharing is allowed for different clients. At runtime the following things are executed: (1) identifying objects by port number (2) communication matching. Thus the port number is dynamically assigned to a channel which has a unique number in the node. The copies of the ROI used in MONARC uses embryo system. This converts OOP language code into C syntax tree. Thus in a way the OOP code gets converted into C equivalent. Thus, with the implementation of OOP, it became possible for completely abstracting server from client and hence enhancing efficiency of programmers at client side.



```
#include "image.pub";
main()
{
    ACTIVE image ed_buf;

    ed_buf=create_active(image, 512, 1024);
    ed_buf=>drawline(4, 4, 4, 1019);
    ed_buf=>drawline(4, 4, 509, 4);
    ...
 }
```

```
defpublic image {
  void init(int, int);
  int drawpoint(int, int);
  int drawline(int, int, int, int);
  int drawcircle(int, int, int);
}

definternal image {
  int width, height;
  unsigned char *body;
  int clip(int, int);
}

defmethod image {
  void init(int w, int h) {
    width=w;
    height=h;
    body=malloc(w*h/8);
  };
  int drawpoint(int x, int y) {
  ....
  };
  ....
}
```

Figure 8: Client Program (left) and Class definition of active object (right)

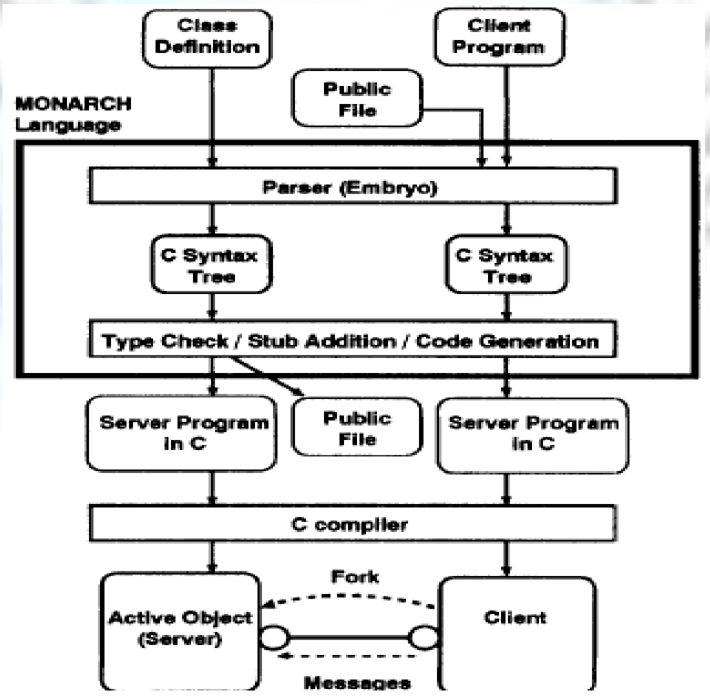


Figure 9: Figure of a compiler process

The biggest advantage of OOP is its capability to transform complex programs into easy solutions. This characteristic is begetted by the fact that OOP supports reusability and separation of functionality. Thus, OOP has been demanded to simplify the control process in automation industry. Already for the same task some high level languages like Petri Hierarchy Net etc. are being used. But in [30]-[34], a similarity was drawn between OOP and Petri Hierarchy (PH) and thus it was projected as to how these existing languages can be replaced by OOP. Because these languages are difficult to be mapped and understand, once written, hence, further enhancements are difficult. Thus, there is a lot of inefficiency and hence, OOP is the need of the hour. Since, there is a similarity between OOP and PH thus, a replacement can be easily provided. In OOP an object has all the characteristics of a class. In a way the various attributes of a class are none other than the characteristics of an object. This object interface is used for communication between systems. Also, there are functional blocks (FB) which are none but predefined classes with already existing functionalities promoting reuse and redefining them at the same time provides case of development. In OOP the entire structure of a system is divided into sub modules independently performing their functions and interlinked. If the lowest module is unable to handle the complexity it communicates the higher modules and in this way the problem is rectified. Similarly, PH is also re-modularization and events. They are discrete events in which the system evolves from one stable state to another because of the existence of an event. In PH, a discrete event system (DES) is modelled by associating states to places and events to transition. With the increase in complexity the use of logic relays needed to be replaced by techniques, more reliable and robust. Hence, even in PH there is hierarchy. The programmer starts with formal methodology either SFC or MOP. HPN (Hierarchy Petri Nets) add aspects that increase their capacity. It allows the scrutiny of behavior, performance, graphic language that determines the behavior of automation. Also HPN allows the modelling of state, events and connections between algorithm and environment.

Hence, the OOP and PH have relationship or communication capabilities. Since OOP has the level distinction of functions, ease of evolvement and similarities with PH, hence, it should be made pervasive in automation. The inclusion will open new possibilities of enhancement which are not possible by the use of PH and other High level Language.

References

[1]. Su Jian, Weng Wengyong, Wang Zebing, “A Teaching Path for Java Object Oriented Programming”, 978-0-7695-3600-2/09 $25.00 © 2009 IEEE DOI 10.1109/IFITA.2009.229.

[2]. B. Joshua, Effective Java, Addison-Wesley, 2001.

[3]. B. Eckel, Thinking in Java, Available online at “http://www.mindview.net/Books/TIJ/”, 2007

[4]. Nelishia Pillay, “Solving Programming Problems in Intelligent Programming Tutors for Teaching the Object-Oriented Programming Paradigm”, 0-7695-0653-4/0$01 0.00 0 2000 IEEE

[5]. W.L. Johnson, “Understanding and Debugging Novice Programs”, in Artificial Intelligence and Learning Environments edited by W.J. Clancy, E. Soloway, MIT Press, 1990, pp. 51-97.

[6]. J. R. Anderson, Reiser B.J., “The Lisp Tutor”, BYTE, April1985,pp. 159 -175.

[7]. Yoke Seng Wong, Dr-Ing Maizatul Hayati Mohamad Yatim, Dr. Wee Hoe Tan, “Use computer game to learn Object-Oriented Programming in computer science courses”, 978-1-4799-3190-3/14/$31.00 ©2014 IEEE

[8]. A. N. Alias Daud Zainab, , and A. B. Zaitun, . “The impact of IT onhigher education for Malaysia.” Presidents’ Forum of Southeast Asia and Taiwan Universities, 2003

[9]. José A. Gallud, Ricardo Tesoriero and Pedro González, “Smalltalk: the Leading Language to Learn Object-Oriented Programming”, 978-83-60810-48-4/$25.00 c 2012 IEEE

[10]. T. Budd. “Object-Oriented Programming”. Addison-Wesley.

[11]. ZuoWen Jiang, ZhiPing Qin, YiMing Zhao, XueMing Lin, “Coding Standard Based Object Oriented Programming Course Teaching Reform*”, 978-0-7695-5004-6/13 $26.00 © 2013 IEEE DOI 10.1109/ICCIS.2013.494

[12]. WeiZhong Shao and FuQingYang, Object oriented Analysis. TsingHu university Publish house, 2006

[13]. Nazlia Omar, Nomariani A.Razik, “Determining the Basic Elements of Object Oriented Programming Using Natural Language Processing”, 978-1-4244-2328-6/08/$25.00 © 2008 IEEE

[14]. J.F. Baldwin and T. P. Martin, “Fuzzy Classes in Object-Oriented Logic Programming”, 0-7803-3645-3/96 $5.0001996 IEEE

[15]. Graham, I., Object-Oriented Methods. 2nd ed. 1994, U.K.: Addison Wesley. 473

[16]. Bob Newman and Martin Payne, “Integration of Object Oriented and Concurrent Programming”, 0-8186-6430-4/94 $4.00 Q 1994 IEEE

[17]. S~~OUSNB.P (1991) The C++ Programming Language, second, edition. Addison Wesley.

[18]. Hon Wai Chun, “A Methodology for Object-Oriented Constraint Programming”

[19]. [BOOC94] G. Booch, Object-oriented Analysis and Design with Applications, 2”d ed., The Benjamid Cummings Publishing Company, Inc., 1994

[20]. WANG Kan, ZHENG YujunUsing LINQ as an Instructional Bridge Between Object-Oriented and Database Programming”, 978-1-4244-3521-0/09/$25.00 ©2009 IEEE

[21]. B. J. Cox, Object-Oriented Programming: An Evolutionary Approach, Reading MA: Addison-Wesely, 1991.

[22]. B. Nicolai, “Is Database Curriculum Information Systems or Information, Technology: An Accreditation Dilemma,” J. Inform. Syst. Educ., vol. 4,pp. 1-11, March 2006.

[23]. Tan Hui1, Wang Feng-Lan2, Li Jing 3, Chen Han-Ming 4, “Software Framework of EMC Test Based on Object-Oriented Programming Idea”, 978-1-4244-7368-7/10/$26.00 ©2010IEEE

[24]. Tang Shi-Ping, Lv Yun-Xi. Automatic test software development platform and application for electromagnetic Compatibility. Safety an EMC, 2005, 4, 58-61.

[25]. C.K. Leung' h R. Leonard, "DEVELOPING A COMPANY WIDE ESTIMATING, DESIGN AND INSTALLATION SYSTEM {PROTECT} USING OBJECT-ORIENTED PROGRAMMING

[26]. L.S. Myint, R. Leonard and R.A. Whiteley. "A methodology for limiting the effects of Halon". Proc. European Trade & Tech. Conf. 1990. pp 158-162.

[27]. Kei Yuasa and Pradeep Kumar Sinha, “NETWORK PROGRAMMING SUPPORT WITH OBJECT-ORIENTED PARADIGM, 0-7803-0922-7/93$03.00 1993QIEEE

[28]. Ananda A. L. et al., .4 Survey of Asynchronous Remote Procedure Calls, ACM Operating Sys- 1.ems Review, Vol. 26, No. 2, pp. 92-109, April 1992

[29]. Microsoft FORTRAN Ver 5.0 Reference Manual. MICROSOFT CORPORATION, 1987

[30]. A. Moreno Reyna. Author1, A. Gómez Ortega1, N. Sierra Romero2, D. Armando Díaz1, S. E. Fernández Murillo1, “Object-Oriented Programming as an alternative to industrial CONTROL”.

[31]. R. Patiño, Programación Orientada a Objetos en Lenguajes No Orientados a Objetos:C, una esperiencia. Scientia et Technica, Vol XI, Nu 29, 2005 pp 107-111.

[32]. C.D. Nielsen, Evaluation of Function Blocks for Asynchronous Design. Department of Computer Science, Technical University of Denmark

[33]. G. Zapata, E. Carrasco, Estructuras Generalizadas para Controladores Logicos Modeladas Mediante Redes de Petri. IEEE Press 2002

[34]. R. Carretero, J. I. Arnesto (2011, May). Integracion de la program macionorientadaaobjectos en la tercera edicion Del estander IEC.